2

NSWC TR 89-33

SOFTWARE SYSTEMS SAFETY DESIGN GUIDELINES AND RECOMMENDATIONS

BY MICHAEL L. BROWN
PROTECTION SYSTEMS DEPARTMENT

MARCH 1989

NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC TR 89-33	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Software Systems Safety Design Guidelines and Recommendations		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael L. Brown		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Warfare Center (H12) Dahlgren, Virginia 22448-5000		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 17X4912.3789 WU # 7CHVSWS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Surface Warfare Center Dahlgren, Virginia 22448-5000		12. REPORT DATE March 1989 13. NUMBER OF PAGES 29
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Space and Naval Warfare Systems Command (SPAWAR-30) Washington, DC 20362-5100		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Safety, Software Safety, Design Guidelines, System Safety, Safety Engineering, Software Design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Software Systems Safety is a discipline within System Safety concerned with the potential safety risks associated with software and computers in safety critical applications. This technical report provides guidelines and recommendations that may be useful in reducing the safety risk of software in safety critical applications. However, it is important to note that these guidelines and recommendations must be tailored to the specific application and must be applied as part of a comprehensive system safety program.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

EXECUTIVE SUMMARY

Software Systems Safety is a discipline within System Safety concerned with the potential safety risks associated with software and computers in safety critical applications. In recent years, a number of mishaps, some resulting in the death of or serious injury to people, have been attributed software errors. Yet, every day new systems come on the market that employ computers for control of safety critical functions and there seems to be no slowing of the trend. Many of the safety critical errors found in software systems are design errors, in other words, the software control of a system is inherently unsafe. Therefore, a significant portion of the Software Systems Safety effort is focused on eliminating design errors and the development of specific safety design requirements that become a part of the final product. The intent of this technical report is to provide some guidelines and recommendations that may be useful in reducing the residual safety risk associated with software controlled systems. However, it is important to note that they must be tailored to the specific application and must be applied as part of a comprehensive system safety program.

[illegible] SYSTEM ENGINEERING [illegible] APPLICATIONS

FOREWORD

This technical report is the result of collecting data related to Software Systems Safety over the last several years. Funding for this and related Software Systems Safety efforts has been provided by the Protection Systems Department and has been coordinated with the Warfare Systems Architecture and Engineering (WSA&E) program at the Space and Naval Warfare Systems Command.

The design guidelines and recommendations in this report come from a variety of sources throughout the Department of Defense, the Defense and Aerospace Industries, and the Food and Drug Administration. Notable contributions have come from Major Bruce Bonnett, formerly with the Air Force Inspection and Safety Center, Norton Air Force Base, California; Dr. M. Frank Houston, Food and Drug Administration; Dr. Nancy Leveson of the University of California at Irvine; and Dr. Peter Neumann of Stanford Research International, Menlo Park, California. Major Bonnett established the Triservice Software Systems Safety Working Group which has been the focal point for much of the work that led to this report. Dr. Houston has the dubious position of trying to cope with the problem of software safety in medical devices.

This report has been reviewed by Dr. R. W. Mattozzi, Systems Engineering Branch; Mr. Michael Ramsburg, Head, Risk Assessment Branch; Mr. F. B. Sanchez, Head, System Safety/Security Division; and Mr. Roy Shank, Protection Systems Department.

Approved by:

M. J. Tino

M. J. Tino, Head
Protection Systems Department

CONTENTS

CHAPTER 1

INTRODUCTION

SOFTWARE SYSTEMS SAFETY

Software Systems Safety is concerned with the potential safety risks associated with software and computers in safety critical system applications such as weapon, fire control and guidance systems, robotics, medical devices, etc. Software Systems Safety is a discipline within System Safety and is the logical extension of System Safety into the software and computer control of the system. In recent years, a number of mishaps, some resulting in death of or serious injury to people and damage to other systems and the environment, have been attributed failures of software. Software errors in a radiation therapy machine allowed the machine to deliver lethal overdoses of radiation to several patients, five of whom died. An error in the design of a blood databank program allowed over 1000 pints of blood that may have been contaminated with the Acquired Immune Deficiency Syndrome (AIDS) to be distributed. A worker in Ford's Michigan Casting Center was struck and killed by a robot due to an error in the system design and its inability to cope with sensor failures. Every day, we see an increase in systems that are employing computers for control of safety critical functions. Weapon systems are now in operation in which the software has full control over the entire engagement process, from detection to kill assessment. Canada will soon commission the first nuclear power plant in North America with a software controlled SCRAM system. The SCRAM system determines when conditions are such that the nuclear reaction should be shut down. Similar systems will likely make their appearance in US nuclear power plants in the not too distant future.

Most of the safety critical errors found in software systems are design errors from a variety of sources. Lack of understanding of how a system is to be used, basic errors in assumptions of how a system or hardware works, ambiguous design requirements and subsequent ambiguity in the implementation, etc. are all a part of the problem. Therefore, a significant portion of the Software Systems Safety effort is focused on eliminating design errors and the development of specific safety design requirements that become a part of the final product.

PURPOSE

The purpose of this technical report is to provide guidelines, recommendations, and other considerations for the design and development of software for systems in which that software has potentially safety critical applications. These guidelines, recommendations, and requirements are designed such that, if properly implemented, they will enhance the safety of software. Safety critical applications of software include not only control functions where the software exercises direct control over hardware, but applications wherein the software-derived data are used to make safety critical decisions or safety critical data are stored. Examples of the latter two issues include structural design programs, monitors of safety critical functions, and extend even to database applications such as medical records. These guidelines are not intended to be used as a checklist but as an augmentation to software safety analyses such as those recommended in the Task 300 series of MIL-STD-882B.

IMPACT OF GUIDELINES AND RECOMMENDATIONS

The guidelines and recommendations contained in this technical report may have some impact on the system and software development processes. However, if the System Safety Program is put in place early, the impact will be minimal and, in fact, may enhance the overall development processes. System safety, as with any other system engineering discipline, requires early integration into the system development process to ensure its maximum benefit and minimal negative impact. Properly done, the safety requirements will have no more impact than any of the other design requirements and will have a high payoff in later stages of the system life cycle.

TAILORING OF DESIGN GUIDELINES AND RECOMMENDATIONS

Many of the guidelines and recommendations contained in this report are written as requirements (i.e., "shall" is used instead of "will" or "should"). The guidelines and recommendations must be tailored to the system or system type under development before inclusion in any contractual or design documents. The user is cautioned to ensure that any guideline or recommendation selected is tailored to the system and that the intent vice the letter of the guideline or recommendation is incorporated in the final design. In some instances, it may be desirable to change "shall" to "will" or "should." Many of the guidelines are common "do's" or "don'ts" of software engineering, however, they are emphasized here to highlight their importance in the area of software systems safety. As with any other source of guidance, the application of these guidelines or recommendations will not provide any assurance that the final system will be safe; they must be applied as part of a comprehensive system safety/software system safety program.

DEFINITIONS

The following definitions are necessary to understand some of the guidelines and recommendations included in this report. They may be used at the discretion of the reader in the preparation of contractual or design documents.

Computer Software Components

Computer Software Components (CSCs) are distinct parts of computer software configuration items (CSCI). CSCs may be further decomposed into other CSCs or Computer Software Units (CSUs). (Reference 2)

Configuration Item

Configuration items are hardware, software, or an aggregation of both, which are designated by the contracting agency for configuration management. (Reference 5)

Safety Critical Computer Software Components

Safety critical computer software components (SCCSCs) are those computer software components (processes, functions, values or computer program states) whose errors (inadvertent or unauthorized occurrence, failure to occur when required, occurrence out of sequence, occurrence in combination with other functions, or erroneous values) can result in a potential hazard, or loss of predictability or control of a system. (Reference 1)

Safety Kernal

A safety kernal is an independent computer program that monitors the state of the system to determine when potentially unsafe system states occur or when transitions to potentially unsafe system states may occur. The safety kernal is designed to prevent the system from entering the unsafe state and return it to a known safe state. Safety kernals may operate in the background as shell programs or in independent computers.

Software Systems Safety

Software Systems Safety is the optimization of System Safety in the design, development, use, and maintenance of software systems and their integration with safety critical hardware systems in an operational environment.

CHAPTER 2

DESIGN GUIDELINES AND RECOMMENDATIONS

GENERAL SYSTEM GUIDELINES

System Design

The system shall be designed for minimum safety risk consistent with mission requirements, mission effectiveness, time, and cost. The order of precedence to be used in the system design to reduce the risk shall be that specified in MIL-STD-882.

Power-Up Tests

A power-up test shall be incorporated in the design that ensures that the system comes up in a safe state, and that safety critical circuits and components are tested to verify their correct operation.

The software design shall ensure that the system is in a safe state during power-up, intermittent faults in the power, or in the event of power loss. The software shall provide for a safe, graceful shutdown of the system due to either a fault or power-down, such that potentially unsafe states are not created.

The software shall be designed to perform a system level check at power-up to verify that the system is safe and functioning properly prior to application of power to safety critical functions including hardware controlled by the software. Periodic tests shall be performed by the software to monitor the safe state of the system.

Primary Control Computer Failure

The system shall be designed such that a failure of the primary control computer will be detected and the system returned to as safe state.

Maintenance Interlocks

Maintenance interlocks shall be provided to preclude hazards to personnel maintaining the system. Where interlocks must be overridden to perform tests, etc., they shall be designed such that they cannot be inadvertently overridden, or left in the overridden state once the system is restored to operational use.

Power Failures

The system shall be designed to provide a safe shut-down under power failure conditions. Fluctuations in power shall not be capable of creating potentially hazardous states.

Work Arounds

The system design shall not permit detected unsafe conditions to be circumvented.

Electromagnetic Radiation/Pulse/Electrostatic Discharge

Computer and external hardware design shall preclude harmful effects from electromagnetic radiation, electromagnetic pulse, or electrostatic interference.

COMPUTERS AND MICROPROCESSORS

Central Processing Units

Central Processing Units (CPUs) that process entire instructions or data words are preferred to those that multiplex data or instructions (e.g., an 8-bit processor is preferred to a 4-bit processor emulating an 8-bit machine).

Microprocessors and computers that can be fully represented mathematically (e.g., the VIPER family of microprocessors) are preferred to those that cannot.

Watchdog Timers

Watchdog timers or similar devices shall be provided to ensure that the microprocessor or computer is operating properly. The timer reset shall be designed such that the software cannot enter an infinite loop and reset the timer as part of the loop sequence.

Memory and Memory Busses

CPUs with separate instruction and data memories and busses are preferred to those using a common data/instruction data buss.

For CPUs using a common data buss, tests shall be conducted to determine the minimum number of clock cycles that must occur between functions on the buss to ensure that invalid information is not picked up by the CPU.

Periodic memory and data buss checks shall be performed. The design of the test sequence shall ensure that single point or likely multiple failures are detected and isolated.

SOFTWARE DESIGN

Design

Software design and code shall be modular. Modules shall have one entry and one exit point in accordance with DoD-STD-2167 design guidelines.

The software shall be designed to detect potentially unsafe conditions and states, either within the software or the overall system, and shall be capable of preventing the potential hazard's occurrence by recovering to a safe state. When a potentially unsafe state has been detected, the software shall alert the operator to the anomaly detected, the action taken, and the safed system configuration and status. If a safety kernal is used to accomplish these tasks, it shall be resident in non-volatile read only memory (ROM).

Analog Function Controls

Software control of analog functions shall have feedback mechanisms that provide positive indications of the function having occurred.

Maintenance

The system and its software shall be designed for ease of maintenance by future personnel not associated with the original design team. The use of techniques for the decomposition of the software system into modules for ease of maintenance is recommended.

Undocumented Features

The operational and support software shall contain only those features and capabilities required by the system. The programs shall not contain "undocumented features".

Safing Systems

The software shall provide for safing hardware subsystems under the control of software when unsafe conditions are detected.

Unauthorized Access

The system design shall prevent unauthorized or inadvertent access to or modification of the software (source or assembly) and object code. This includes preventing self-modification of the code.

Catastrophic and Critical Functions

Functions that are potentially catastrophic or critical shall be controlled by at least two independent functions.

Inadvertent Instruction Jumps

The system shall provide for fail-safe recovery from inadvertent instruction jumps.

The system shall detect inadvertent jumps within or into Safety Critical Computer Software Components, return the system to a safe state, and, if practical, perform diagnostics and fault isolation to determine the cause of the inadvertent jump.

Interrupt Processing

The software shall be capable of discriminating between valid and invalid (e.g., spurious) internal interrupts and shall recover to a safe state if they occur.

Language Constructs

Halt, stop, or wait instructions shall not be used.

GO-TO statements shall not be used.

Flags

Flags shall be unique and shall have a single purpose.

Files

Files shall be unique and shall have a single purpose. Scratch files shall not be used for storing or transferring safety critical information between processes.

Addressing Schemes

Indirect addressing methods shall not be used unless absolutely necessary. When used, the address shall be verified as being within acceptable limits prior to execution.

System Errors

The software shall make provisions for logging all system errors detected.

SPECIFIC GUIDELINES AND RECOMMENDATIONS

Operational Programs

Operational program loads shall not contain unused executable code. Unused executable code shall be removed from the source and the program recompiled.

Operational program loads shall not contain unreferenced variables.

All processor memory not used for or by the operational program shall be initialized to a pattern that will cause the system to revert to a safe state if executed. It shall not be filled with random numbers, halt, stop, wait, or no-operation instructions. Data or code from previous overlays or loads shall not be allowed to remain. (e.g., If the processor architecture halts upon receipt of non-executable code, a watchdog timer shall be provided with an interrupt routine to revert the system to a safe state. If the processor flags non-executable code as an error, an error handling routine shall be developed to revert the system to a safe state and terminate processing.) Information shall be provided to the operator to alert him to the failure and the reversion to a safe system state.

Overlays shall all occupy the same amount of memory. Where less memory is required for a particular function, the remainder shall be initialized to a pattern that will cause the system to revert to a safe state if executed. It shall not be filled with random numbers, halt, stop, no-op or wait instructions or data or code from previous overlays.

SAFETY CRITICAL COMPUTER SOFTWARE COMPONENTS

SCCSCs and other safety critical software items shall not be used in one-to-one assignment statements.

SCCSCs and safety critical interfaces shall be under positive control at all times.

Safety critical timing functions shall be controlled by the computer and shall not rely on human input. Safety critical timing values shall not be modifiable by the operator from system consoles unless required as part of the system's functional capabilities.

Safety critical functions shall be grouped together and the number of affected program modules shall be minimized where possible within the constraints of operational effectiveness, computer resources, and good software design practices.

Conditional statements shall have all possible conditions satisfied and under full software control (i.e., there shall be no unresolved potential input to the conditional statement).

Safety critical functions shall exhibit strong data typing. Safety critical functions shall not employ a logic "1" and "0" to denote the safe and "armed" (potentially hazardous) states. The "armed" and safe states shall be represented by at least a four bit unique pattern. The safe state shall be a pattern that cannot, as a result of a one or two bit error, represent the "armed" pattern. If a pattern other than these two unique codes is detected, the software shall flag the error, revert to a safe state and notify the operator.

Decision statements in safety critical computer software components shall not rely on inputs of all ones or all zeros, particularly when this information is obtained from external sensors.

Operational checks of testable safety critical system elements shall be made immediately prior to performance of a related safety critical operation.

Upon completion of tests wherein safety interlocks are removed, disabled or bypassed, restoration of those interlocks shall be verified by the software prior to being able to resume normal operation. While overridden, a display shall be made on the operator's or test conductor's console of the status of the interlocks.

INTERFACE DESIGN GUIDELINES AND RECOMMENDATIONS

Inter-CPU communications shall successfully pass verification checks in both CPUs prior to data transfer. Periodic checks shall be performed to ensure the validity of data transmissions.

Data transfer messages shall be of a predetermined format and content. Each transfer shall contain a word or character string indicating the type of data and content of the message. As a minimum, parity checks and checksums shall be used for verification of correct data transfer. Character Recognition Codes (CRCs) shall be used where practical.

External functions requiring two or more safety critical signals from the software (e.g., arm and fire) shall not receive all of the necessary signals from a single register or input/output port. In addition, these signals shall not be generated by a single CPU command.

The software shall be capable of discriminating between valid and invalid (e.g., spurious) external interrupts and shall recover to a safe state in the event of an erroneous external interrupt.

Decision statements shall not rely on inputs of all ones or all zeros, particularly when this information is obtained from external sensors.

Safety critical input or output functions shall not employ a logical "1" and "0" to denote the safe and "armed" (potentially unsafe) state. Safety critical functions shall be represented by at least four bits. The "armed" state shall be represented by a unique bit pattern. The safe state shall be represented by another unique pattern that cannot, as a result of a one or two bit error, represent the "armed" pattern. If a code other than these two unique codes is detected, the software shall flag the error, revert to a safe state, and notify the operator of the erroneous input or output.

Feedback loops shall be designed such that the software cannot cause a runaway condition due to the failure of a feedback sensor. Known component failure modes shall be considered in the design of the software.

Input/output registers and ports shall not be used for both safety critical and non-critical functions.

Limit and reasonableness checks shall be performed on all analog and digital inputs and outputs prior to action occurring based on those values.

The software shall be designed to detect failures in external hardware input or output devices and revert to a safe state upon their occurrence. The design shall consider potential failure modes of the hardware involved.

The software shall be designed such that the full scale and zero representations of the software are fully compatible with the scales of any digital to analog, analog to digital, digital to synchro, and/or synchro to digital converters.

OPERATOR INTERFACE DESIGN GUIDELINES AND RECOMMENDATIONS

The software shall be designed such that the operator may cancel current processing with a single action and have the system revert to a safe state. The system shall be designed such that the operator may exit potentially unsafe states with a single action. This action shall revert the system to a known safe state (e.g., the operator shall be able to terminate missile launch processing with a single action. This action shall safe the missile. The action may consist of pressing two keys simultaneously).

Two or more unique operator actions shall be required to initiate any potentially hazardous function or sequence of functions. The actions required shall be designed to minimize the potential for inadvertent actuation.

Operator displays, legends, and other interactions shall be clear, concise, and unambiguous.

The software shall be capable of detecting improper operator entries or sequences of entries or operations. It shall alert the operator to the erroneous entry or operation. Alerts shall indicate the error and corrective action. The software shall also provide positive confirmation of valid data entry or actions taken (i.e., the system shall provide visual and/or aural feedback to the operator such that the operator knows that the system has accepted the action and is processing it. Aural feedback should be audible against expected background noise.) The system shall also provide a real-time indication that it is functioning. Processing functions requiring several seconds or longer shall provide a status indicator to the operator during processing.

Alerts shall be designed such that routine alerts are readily distinguished from safety critical alerts. The operator shall not be able to clear a safety critical alert without taking corrective action or performing subsequent actions required to complete the operation.

IDENTIFICATION OF SAFETY CRITICAL COMPUTER SOFTWARE COMPONENTS

The priority structure of fault detection and safing or correcting logic shall be considered safety critical. Software units or modules handling or responding to these faults shall be designated SCCSCs.

Interrupt processor software, interrupt priority schemes and routines which disable or enable interrupts shall be designated as SCCSCs.

Software generated signals which have autonomous control over hardware shall be designated as SCCSCs.

Software generated signals which have been shown through detailed analyses (e.g., Fault Tree Analyses) to directly influence movement of hardware components or initiate safety critical actions (e.g., rocket motor arm command) shall be designated as SCCSCs.

Software generated outputs that display the status of safety critical hardware systems shall be designated SCCSCs.

GUIDELINES FOR SOFTWARE DEVELOPMENT PHASES

Coding Phase

Desk audits and peer reviews shall be used to verify implementation of design requirements in the source code with particular attention paid to the implementation of identified safety critical computer software components and the guidelines provided in this document.

At least two people shall be thoroughly familiar with the design, code, and operation of each software module in the system.

Configuration control shall be established as soon as a practical software baseline can be established. All subsequent software changes must be approved by the Software Configuration Control Board pricr to their implementation. A member of the Board shall be tasked with the responsibility for evaluation of all software changes for their potential safety impact. This member should be a member of the system safety engineering team. A member of the hardware Configuration Control Board shall be a member of the Software Configuration Control Board to keep members apprised of hardware changes and to ensure that software changes do not conflict with or introduce potential safety hazards due to hardware incompatibilities.

Conditional statements shall be analyzed to ensure that the conditions are reasonable for the task and that all potential conditions are satisfied and not left to a default condition. All condition statements shall be commented with their purpose and expected outcome for given conditions.

Reviews of the software source code shall ensure that the code and comments agree.

Patches shall be prohibited throughout the development process. All software changes shall be coded in the source language and compiled prior to entry into test equipment.

Values for timers shall be commented in with the code. Comments shall include a description of the timer function, its value and the rationale or a reference to the documentation explaining the rationale for the timer value. These values shall be verified and shall be examined for reasonableness for the intended function.

Software Testing Phase

Software testing shall include NO-GO path testing.

Software testing shall include hardware and software input failure mode testing.

Software testing shall include boundary, out-of-bounds, and boundary crossing test conditions.

Software testing shall include inputs values of zero, zero crossing, and approaching zero from either direction.

Software testing shall include minimum and maximum input data rates in worst case configurations to determine the system's capabilities and response to these environments.

SCCSCs in which changes have been made shall be subjected to complete regression testing.

Operator interface testing shall include operator errors during safety critical operations to verify safe system response to these errors.

Maintenance

Changes to safety critical computer software components on deployed or fielded systems shall be issued as a complete package for the modified unit or module and shall not be patched.

Firmware changes shall be issued as a fully functional and tested circuit card. Design of the card and the installation procedures should minimize the potential for damage to the circuits due to mishandling, electrostatic discharge, or normal or abnormal storage environments.

CHAPTER 3

CONCLUSIONS AND RECOMMENDATIONS

Many of the design guidelines and recommendations in this technical report are based on lessons derived from past failures of software controlled systems; others are the result of analyses performed on safety critical systems; still others are simply an extension of good software engineering practices. The guidelines and recommendations should not be used as a checklist for the development of safety critical software. Likewise, it is not sufficient that they be used as a checklist for evaluating a proposal or a final product or system. These guidelines and recommendations must be applied as part of a comprehensive system safety program.

APPENDIX A

REFERENCE DOCUMENTS

The following documents form the basis for or are referenced in this document:

1.	MIL-STD-882	System Safety Program Requirements
2.	DoD-STD-2167	Defense System Software Development
3.	DoD-STD-2168	Defense System Software Quality Program
4.	DoD-HDBK-287	Tailoring DoD-STD-2167A Requirements
5.	DoD-HDBK-480	Configuration Control for Engineering Changes, Deviations and Waivers
6.	DoD-HDBK-483	Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs

Copies of the above documents required by contractors in connection with a specific procurement may be obtained through the procuring activity or as directed by the contracting officer. Government agencies may obtain copies through local technical library services.

Distribution

	Copies
Commander,	
Space and Naval Warfare Systems Command	
Attn: SPAWAR 003-43 (Tony Sliwa)	1
SPAWAR 003-43B (Al Friend)	1
SPAWAR 005-35 (Marianne Vashnav)	1
SPAWAR 3212	1
Washington, DC 20363	
Commander,	
Naval Air Systems Command	
Attn: AIR 516C (J. Gibble)	1
AIR 516C1 (J. Nerrie)	1
Washington, DC 20375-5000	
Commander,	
Naval Sea Systems Command	
Attn: SEA 666 (Ed Kratovil)	1
SEA 6663 (Paul Wright)	1
SEA 92Q41 (Jeff Thomas)	2
Washington, DC 20363	
Commander,	
Naval Safety Center	
Attn: Code 90 (P. Kinzey)	1
Code 94 (W. Mannschreck)	1
Norfolk NAS, VA 23511-5796	
Commander,	
Naval Weapons Center	
Attn: Code 3687 (Jerry Mathre)	2
China Lake, CA 93555-6001	
Commander,	
Naval Research Laboratory	
Attn: Code 5540B (H.O. Lubbes)	1
Code 7593 (Dr. Landwehr)	1
Washington, DC 20375-5000	
Commander,	
Naval Air Engineering Center	
Attn: Code 9314	1
Lakehurst, NJ 08733-5100	
Commander,	
David Taylor Naval Research Center	
Attn: Code 1202 (Steve Cohen)	1
Bethesda, MD 20084	

Distribution
(Continued)

	Copies
Commander,	
Naval Air Development Center	
Attn: Code 7022 (John Pendergast)	1
Warminster, PA 18974-5000	
Commander,	
Naval Underwater Systems Center	
Attn: Code 43 (Jerry Morris)	1
Code 43 (Gerry Lipsett)	1
Newport, RI 02840	
Commander,	
Naval Ocean Systems Center	
Attn: Code 921 (Edward Glunt)	1
Code 921 (Leroy Haulsey)	1
San Diego, CA 92152-5000	
Commander,	
Naval Coastal Systems Center	
Attn: Code 5320 (Tom English)	1
Panama City, FL 32407	
Commander,	
US Army Communications Engineering Command	
Attn: AMSEL-SF-SEP (Rod Murphy)	1
Fort Monmouth, NJ 07703	
Director	
US Army Aviation Systems Command	
Attn: AMSAV-XAP (Jose Caraballo)	1
AMSAV-W, (Dr. Vernon Allen)	1
4800 Goodfellow Blvd.	
St. Louis, MO 63120-1798	
Commanding General,	
US Army Armament and Material Division	
Attn: AMSMC-QAH-A(D) (Paul Janusz)	1
Bldg. 62	
Dover, NJ 07801-5001	
US Army Laboratory Command	
Harry Diamond Laboratories	
2800 Powder Mill Road	
Attn: SLCHD-TA (Dave Overman)	1
Adelphi, MD 20783-1197	

Distribution (Continued)

Copies

Program Executive Officer
Program Manager for Chemical Demilitarization
PEO-PMCML-DEMIL
Attn: ACM-PEO-CDS 1
Aberdeen Proving Ground, MD 21010

Director,
US Army Human Engineering Laboratory
Attn: Richard Armstrong 1
Box 716
Fort Rucker, AL 36362

Headquarters,
Air Force Systems Command
Attn: AFSC-HQ (Harris Yeager) 1
Andrews AFB, MD 20334

Director
Air Force Systems Command
Space Division
Attn: SD/SE (Dr. Louis Huang) 1
SD/SE (Roger Lockwood) 1
P.O. Box 92960
Los Angeles, CA 90009-2960

Commanding Officer
Wright Patterson Aeronautics Laboratory
Attn: AFWAL-SES (Randy Janssen) 1
Wright Patterson AFB, OH 45433

Headquarters,
Air Force Inspection and Safety Center
Attn: AFISC/SE 1
AFISC/SESD 2
Norton AFB, CA 92409-7001

Headquarters
Air Force Operational Test and Evaluation Command
Attn: AFOTEC/SE Capt. Steven Mattern 2
Kirtland AFB, NM 87117-7001

Headquarters,
National Aeronautics and Space Administration
Attn: Code QD (Pete Rutledge) 1
Washington, DC 20546

Distribution
(Continued)

Copies

Director,
Food and Drug Administration
Center for Radiological Devices and Health
Attn: Code HFZ150 (Frank Houston) 2
5600 Fishers Lane
Rockville, MD 20857

National Institute of Standards and Technology
Institute for Computer Sciences and Technology
Attn: Dolores Wallace 1
Technology Bldg. Rm B266
Gaithersburg, MD 20899

General Dynamics
Valley Systems Division
P.O. Box 50-800, MS 601-21
Attn: Robert Sweginnis 1
Ontario, CA 91761-1085

Applied Ordnance Technology, Inc.
Attn: Edward Daugherty 1
Joel Pulliam 1
Suite 909
2001 Jefferson Davis Hwy.
Arlington, VA 22202

General Dynamics
Fort Worth Division
Attn: MZ 2288 (Kevin Martin) 1
PO Box 748
Fort Worth, EX 76101

Boeing Aerospace Co.
P.O. Box 3999
Attn: MS 1E-62 (Irving Meyerson) 1
Seattle, WA 98124-2499

Ketron, Inc.
Attn: Gerald Donovan 1
600 Louis Drive
Suite 203
Warminster, PA 18974

TRW, Inc.
Attn: Carl Hocevar 1
One Space Park
Redondo Beach, CA 90278

Distribution
(Continued)

Copies

Dr. Nancy Leveson 1
University of California at Irvine
Information and Computer Sciences Department
Irvine, CA 92717

Stanford Research International
Attn: Dr. Peter Neumann 1
BN 168
333 Ravenswood
Menlo Park, CA 94025-3493

Internal Distribution:

E211 (Sullivan)	1
E231	10
G44 (Munach)	1
G44 (Bagnell)	1
H	1
H02	1
H10	1
H101	1
H11	10
H12	1
H13	2
H14	1
N06	2
N42 (Akin)	1